4. 테스팅과 예외처리

stdout으로 보낸 결과물 테스팅

```
import unittest
import errno

def parse_int(s):
    return int(s)

class TestConversion(unittest.TestCase):
    def test_bad_int(self):
        self.assertRaises(ValueError, parse_int, "N/A")

    def test_bad_int_msg(self):
        self.assertRaisesRegex(ValueError, 'invalid literal .*', parse_int, 'N/A')

class TestIO(unittest.TestCase):
    def test_file_not_found(self):
        try:
            f = open('/file/not/found')
        except IOError as e:
            self.assertEqual(e.errno, errno.ENOENT)
        else:
            self.fail("IOError not raised")
```

```
if __name__ == '__main__':
    unittest.main(argv=['first-arg-is-ignored'], exit=False)
```

```
.........s.sx
----------------------------------------------------------------------
Ran 14 tests in 0.007s

OK (skipped=2, expected failures=1)
```

"유닛 테스트를 작성 중인데, 선택한 객체에 패치를 적용해서 유닛 테스트에서 어떻게 사용하는지 확인하고 싶다.(예: 특정 파라미터와 함께 호출되었는지, 선택한 속성에 접근했는지 등)"

"이 문제를 해결하는데 unittest.mock.patch() 함수가 도움이 된다. patch()는 홀로 사용할 수도 있고 데코레이터, 콘텍스트 매니저로도 사용 가능하다."

유닛 테스트에서 예외 조건 시험하기

```
from io import StringIO
from unittest import TestCase
from unittest.mock import patch

def urlprint(protocol, host, domain):
    url = '{}://{}.{}'.format(protocol, host, domain)
    print(url)

class TestURLPrint(TestCase):
    def test_url_gets_to_stdout(self):
        protocol = 'http'
        host = 'www'
        domain = 'example.com'
        expected_url = '{}://{}.{}\n'.format(protocol, host, domain)

        with patch('sys.stdout', new=StringIO()) as fake_out:
            urlprint(protocol, host, domain)
            self.assertEqual(fake_out.getvalue(), expected_url)
```

"올바른 입력이 주어졌을 때 올바른 결과가 나오는지 코드를 검증 하고 싶다."

유닛 테스트에서 예외 조건 시험하기

```
if __name__ == '__main__':
    import unittest
    unittest.main(argv=['first-arg-is-ignored'], exit=False)
```

```
.........s.sx
----------------------------------------------------------------------
Ran 14 tests in 0.009s

OK (skipped=2, expected failures=1)
```

"unittest.mock 모듈의 patch() 함수를 사용하면 테스팅 용도로 sys.stdout을 흉내 내고 다시 제자리에 돌려 놓을 수 있다."

파일에 테스트 결과 기록

```
import unittest
import errno

def parse_int(s):
    return int(s)

class TestConversion(unittest.TestCase):
    def test_bad_int(self):
        self.assertRaises(ValueError, parse_int, "N/A")

    def test_bad_int_msg(self):
        self.assertRaisesRegex(ValueError, 'invalid literal .*', parse_int, 'N/A')

class TestIO(unittest.TestCase):
    def test_file_not_found(self):
        try:
            f = open('/file/not/found')
        except IOError as e:
            self.assertEqual(e.errno, errno.ENOENT)
        else:
            self.fail("IOError not raised")
```

파일에 테스트 결과 기록

```
import sys
def main(out=sys.stderr, verbosity=2):
    loader = unittest.TestLoader()
    suite = loader.loadTestsFromModule(sys.modules[__name__])
    unittest.TextTestRunner(out, verbosity=verbosity).run(suite)

if __name__ == '__main__':
    with open('testing.out', 'w') as f:
        main(f)
```

"유닛 테스트 결과를 화면에 출력하지 않고 파일에 기록하고 싶다."

테스트 실패 예측과 건너뜀

```
import unittest
import os
import platform

class Tests(unittest.TestCase):
    def test_0(self):
        self.assertTrue(True)

    @unittest.skip('skipped test')
    def test_1(self):
        self.fail("should have failed!")

    @unittest.skipIf(os.name=='posix', 'Not supported on Unix')
    def test_2(self):
        import winreg

    @unittest.skipUnless(platform.system() == 'Darwin', 'Mac specific test')
    def test_3(self):
        self.assertTrue(True)

    @unittest.expectedFailure
    def test_4(self):
        self.assertEqual(2+2, 5)
```

```
if __name__ == '__main__':
    unittest.main(argv=['first-arg-is-ignored'], exit=False, verbosity=2)
```

"유닛 테스트 내부에 실패할 것이라 예상되는 테스트를 선택하거나 건너뛰고 싶다."

"unittest 모듈에는 선택한 테스트 모듈에 적용해서 그 처리를 제어할 수 있는 데코레이터가 있다."

```
test_add (__main__.MyCalcTest) ... ok
test_substract (__main__.MyCalcTest) ... ok
test_bad_int (__main__.TestConversion) ... ok
test_bad_int_msg (__main__.TestConversion) ... ok
test_file_not_found (__main__.TestIO) ... ok
test_isupper (__main__.TestStringMethods) ... ok
test_split (__main__.TestStringMethods) ... ok
test_upper (__main__.TestStringMethods) ... ok
test_url_gets_to_stdout (__main__.TestURLPrint) ... ok
test_0 (__main__.Tests) ... ok
test_1 (__main__.Tests) ... skipped 'skipped test'
test_2 (__main__.Tests) ... ok
test_3 (__main__.Tests) ... skipped 'Mac specific test'
test_4 (__main__.Tests) ... expected failure

----------------------------------------------------------------------
Ran 14 tests in 0.008s

OK (skipped=2, expected failures=1)
```

4. 테스팅과 예외처리

4.1 파이썬 유닛 테스트

4.2 파이썬 예외 처리

다중 예외 처리


```
try:
    #10 * (1/0)
    4 + spam*3
except ( ZeroDivisionError, NameError ):
    print('예외발생')
```

```
예외발생
```

```
try:
    #10 * (1/0)
    '1' + 1
except ( ZeroDivisionError, NameError ):
    print('예외발생')
except ( TypeError ):
    print('타입 예외발생')
```

```
타입 예외발생
```

다중 예외 처리


```
try:
    f = open("noname")
except ( FileNotFoundError, PermissionError ):
    print('파일 예외발생')
```

```
파일 예외발생
```

```
try:
    f = open("noname")
except OSError :
    print('파일 예외발생')
```

```
파일 예외발생
```

```
import errno
try:
    f = open("noname")
except OSError as e:
    if e.errno == errno.ENOENT:
        print('File not found')
    elif e.errno == errno.EACCES:
        print('Permission denied')
```

```
File not found
```

```
print(errno.errorcode)
```

```
{19: 'ENODEV', 10065: 'WSAEHOSTUNREACH', 122: 'ENOMSG', 120: 'ENODATA', 40:
'ENOSYS', 32: 'EPIPE', 22: 'EINVAL', 132: 'EOVERFLOW', 4: 'EINTR', 10068:
'WSAEUSERS', 41: 'ENOTEMPTY', 10055: 'WSAENOBUFS', 134: 'EPROTO', 10071:
'WSAEREMOTE', 10: 'ECHILD', 10062: 'WSAELOOP', 18: 'EXDEV', 7: 'E2BIG', 3:
'ESRCH', 10040: 'WSAEMSGSIZE', 10047: 'WSAEAFNOSUPPORT', 10064:
'WSAEHOSTDOWN', 10046: 'WSAEPFNOSUPPORT', 10042: 'WSAENOPROTOOPT',
...
```

다중 예외 처리

```
import errno
try:
    f = open("noname")
except OSError:
    print('It failed')
except FileNotFoundError:
    print('File not found')
```

```
It failed
```

```
FileNotFoundError.__mro__
```

```
(FileNotFoundError, OSError, Exception, BaseException, object)
```

모든 예외 캐치


```
try:
    f = open("noname")
except Exception as e:
    print('Reason:', e)
```

```
Reason: [Errno 2] No such file or directory: 'noname'
```

```
def parse_int(s):
    try:
        n = int(v)
    except Exception:
        print("Couldn't parse")

parse_int('n/a')
```

```
Couldn't parse
```

```
parse_int('42')
```

```
Couldn't parse
```

모든 예외 캐치


```
def parse_int(s):
    try:
        n = int(v)
    except Exception as e:
        print("Couldn't parse")
        print('Reason:', e)

parse_int('42')
```

```
Couldn't parse
Reason: name 'v' is not defined
```

커스텀 예외 생성


```
class CustomError(Exception):
    def __init__(self, message, status):
        super().__init__(message, status)
        self.message = message
        self.status = status

try:
    raise CustomError("It failed", 33)
except CustomError as e:
    print('Reason:', e.args)
```

```
Reason: ('It failed', 33)
```

```
try:
    raise RuntimeError("It failed")
except RuntimeError as e:
    print('Reason:', e.args)
```

```
Reason: ('It failed',)
```

커스텀 예외 생성

```
try:
    raise RuntimeError("It failed", 42, "spam")
except RuntimeError as e:
    print('Reason:', e.args)
```

```
Reason: ('It failed', 42, 'spam')
```

다른 예외에 대한 응답으로 예외 발생

```
def example1():
    try:
        int('N/A')
    except ValueError as e:
        raise RuntimeError('A parsing error occurred') from e

def example2():
    try:
        int('N/A')
    except ValueError as e:
        print('It failed. Reason:', e)

def example3():
    try:
        int('N/A')
    except ValueError as e:
        raise RuntimeError('A parsing error occurred') from None
```

다른 예외에 대한 응답으로 예외 발생

```
import traceback
try:
    example1()
except Exception:
    traceback.print_exc()
```

```
Traceback (most recent call last):
  File "<ipython-input-41-b0aff642396d>", line 3, in example1
    int('N/A')
ValueError: invalid literal for int() with base 10: 'N/A'

The above exception was the direct cause of the following exception:

Traceback (most recent call last):
  File "<ipython-input-42-598a41abc739>", line 3, in <module>
    example1()
  File "<ipython-input-41-b0aff642396d>", line 5, in example1
    raise RuntimeError('A parsing error occurred') from e
RuntimeError: A parsing error occurred
```

다른 예외에 대한 응답으로 예외 발생


```
try:
    example2()
except Exception:
    traceback.print_exc()
```

```
It failed. Reason: invalid literal for int() with base 10: 'N/A'
```

```
try:
    example3()
except Exception:
    traceback.print_exc()
```

```
Traceback (most recent call last):
  File "<ipython-input-44-a0f1927aba52>", line 2, in <module>
    example3()
  File "<ipython-input-41-b0aff642396d>", line 17, in example3
    raise RuntimeError('A parsing error occurred') from None
RuntimeError: A parsing error occurre
```

마지막 예외 다시 발생

```
def example():
    try:
        int('N/A')
    except ValueError:
        print("Didn't work")
        raise

example()
```

```
Didn't work
...
ValueError: invalid literal for int() with base 10: 'N/A'
```

```
import time
from functools import wraps

def timethis(func):
    @wraps(func)
    def wrapper(*args, **kwargs):
        start = time.perf_counter()
        r = func(*args, **kwargs)
        end = time.perf_counter()
        print('{}.{} : {}'.format(func.__module__, func.__name__, end-start))
        return r
    return wrapper

if __name__ == '__main__':
    @timethis
    def countdown(n):
        while n > 0:
            n -= 1

    countdown(10000000)
```

```
__main__.countdown : 0.467705199996999
```

```
from contextlib import contextmanager

@contextmanager
def timeblock(label):
    start = time.perf_counter()
    try:
        yield
    finally:
        end = time.perf_counter()
        print("{} : {}".format(label, end-start))

with timeblock("counting"):
    n = 10000000
    while n > 0 :
        n -= 1
```

```
counting : 0.7532543000052101
```